

Transkribus API Integration



I need to create a API method for Transkribus API using Metagrapho Documentation.

Steps:

1. Obtain an Access Token from Transkribus Metagrapho API <https://transkribus.eu/processing/v1/>
2. Submit data for processing, pass image from "input_folder", as base64 with given layout information for text recognition. This is the response: {

```
"processId": 47725,
"status": "CREATED"
}
```

We use this process id later

3. Retrieve processing status and result

Get the processing status of a previous post identified by its processId

I want all this code inside this method

```
def call_transkribus_api(image_path):
    # Placeholder for actual API call
    return "recognized text transkribus"
```

To implement the `call_transkribus_api(image_path)` method for processing an image using the Transkribus Metagrapho API, follow these steps:

1. **Obtain an Access Token:** Authenticate with the Transkribus API using your username and password to receive an access token.
2. **Submit the Image for Processing:** Encode the image in base64 and submit it to the API along with the required layout and HTR model IDs.
github.com+9
3. **Retrieve Processing Status and Result:** Poll the API using the returned `processId` until the processing is complete, then retrieve the recognized text.

Here's how you can implement this in Python:

```
python

import base64
import requests
import time

def call_transkribus_api(image_path):
    # Replace with your actual Transkribus credentials
    USERNAME = 'your_username'
    PASSWORD = 'your_password'
    CLIENT_ID = 'processing-api-client'

    # Model IDs (replace with appropriate IDs for your use case)
    LINE_MODEL_ID = 49272 # Example: Mixed Text Line Orientation
    HTR_MODEL_ID = 51170 # Example: Transkribus Print M1

    # Step 1: Obtain Access Token
```

```

token_url = 'https://account.readcoop.eu/auth/realms/readcoop/protocol/openid-connect/token'
token_data = {
    'grant_type': 'password',
    'username': USERNAME,
    'password': PASSWORD,
    'client_id': CLIENT_ID
}
token_headers = {'Content-Type': 'application/x-www-form-urlencoded'}
token_response = requests.post(token_url, data=token_data, headers=token_headers)
token_response.raise_for_status()
access_token = token_response.json()['access_token']

# Step 2: Submit Image for Processing
with open(image_path, 'rb') as image_file:
    image_bytes = image_file.read()
image_base64 = base64.b64encode(image_bytes).decode('utf-8')

process_url = 'https://transkribus.eu/processing/v1/processes'
process_headers = {
    'Authorization': f'Bearer {access_token}',
    'Content-Type': 'application/json'
}
process_payload = {
    'image': image_base64,
    'lineDetectionModel': LINE_MODEL_ID,
    'htrModel': HTR_MODEL_ID
}
process_response = requests.post(process_url, json=process_payload, headers=process_headers)
process_response.raise_for_status()
process_id = process_response.json()['processId']

# Step 3: Poll for Processing Status and Retrieve Result
status_url = f'https://transkribus.eu/processing/v1/processes/{process_id}'
while True:
    status_response = requests.get(status_url, headers={'Authorization': f'Bearer {access_token}'})
    status_response.raise_for_status()
    status_data = status_response.json()
    status = status_data.get('status')
    if status == 'FINISHED':
        recognized_text = status_data.get('result', {}).get('text', '')
        return recognized_text
    elif status == 'FAILED':
        raise Exception('Processing failed.')
    time.sleep(5) # Wait before polling again

```

Notes:

- Ensure you replace 'your_username' and 'your_password' with your actual Transkribus credentials.
- The **LINE_MODEL_ID** and **HTR_MODEL_ID** should correspond to the models you intend to use. You can find available models and their IDs in the Transkribus WebApp or documentation.
github.com+1
- The polling interval is set to 5 seconds; adjust this as needed based on expected processing times.
- Handle exceptions and errors appropriately in your production code to ensure robustness.

This implementation follows the standard procedure for interacting with the Transkribus Metagrapho API as outlined in their documentation.



```

def submit_image_by_base64(self, base64_string, htr_id, layout_content=None):
    url = f"{self.api_url}/processes"
    headers = {

```

```
'Authorization': f'Bearer {self.access_token}',
'Content-Type': 'application/json',
'Accept': 'application/json'
}

payload = {
    "config": {
        "textRecognition": {
            "htrId": htr_id
        }
    },
    "image": {
        "base64": base64_string
    }
}

# Optionally include layout information if provided
if layout_content:
    payload["content"] = layout_content
else:
    payload["config"]["lineDetection"] = {
        "modelId": 51962 # Default line detection model ID
    }

response = requests.post(url, json=payload, headers=headers)
if response.status_code == 200:
    data = response.json()
    print(f"Image submitted successfully via base64! Process ID: {data['processId']}")
    return data['processId']
else:
    print(f"Base64 image submission failed: {response.status_code} - {response.text}")
    return None
```

Thank you for sharing your `submit_image_by_base64` method. It effectively handles the submission of base64-encoded images to the Transkribus Metagrapho API. Building upon this, you can integrate the entire process—including authentication, image submission, and result retrieval—into a single method. Here's how you can structure the `call_transkribus_api` function:

`call_transkribus_api(image_path)` Implementation

```
python

import base64
import requests
import time

def call_transkribus_api(image_path):
    # Replace with your actual Transkribus credentials
    USERNAME = 'your_username'
    PASSWORD = 'your_password'
    CLIENT_ID = 'processing-api-client'
```

```
# Model IDs (replace with appropriate IDs for your use case)
LINE_MODEL_ID = 51962 # Default line detection model ID
HTR_MODEL_ID = 51170 # Example: Transkribus Print M1

# Step 1: Obtain Access Token
token_url = 'https://account.readcoop.eu/auth/realms/readcoop/protocol/openid-connect/token'
token_data = {
    'grant_type': 'password',
    'username': USERNAME,
    'password': PASSWORD,
    'client_id': CLIENT_ID
}
token_headers = {'Content-Type': 'application/x-www-form-urlencoded'}
token_response = requests.post(token_url, data=token_data, headers=token_headers)
token_response.raise_for_status()
access_token = token_response.json()['access_token']

# Step 2: Encode Image to Base64
with open(image_path, 'rb') as image_file:
    image_bytes = image_file.read()
image_base64 = base64.b64encode(image_bytes).decode('utf-8')

# Step 3: Submit Image for Processing
process_url = 'https://transkribus.eu/processing/v1/processes'
process_headers = {
    'Authorization': f'Bearer {access_token}',
    'Content-Type': 'application/json',
    'Accept': 'application/json'
}
process_payload = {
    "config": {
        "textRecognition": {
            "htrId": HTR_MODEL_ID
        },
        "lineDetection": {
            "modelId": LINE_MODEL_ID
        }
    },
    "image": {
        "base64": image_base64
    }
}
process_response = requests.post(process_url, json=process_payload, headers=process_headers)
process_response.raise_for_status()
process_id = process_response.json()['processId']

# Step 4: Poll for Processing Status and Retrieve Result
status_url = f'https://transkribus.eu/processing/v1/processes/{process_id}'
while True:
    status_response = requests.get(status_url, headers={'Authorization': f'Bearer {access_token}'})
    status_response.raise_for_status()
    status_data = status_response.json()
    status = status_data.get('status')
    if status == 'FINISHED':
        recognized_text = status_data.get('result', {}).get('text', '')
        return recognized_text
    elif status == 'FAILED':
        raise Exception('Processing failed.')
    time.sleep(5) # Wait before polling again
```

Notes:

- **Credentials:** Ensure you replace 'your_username' and 'your_password' with your actual Transkribus credentials.
- **Model IDs:** The `LINE_MODEL_ID` and `HTR_MODEL_ID` should correspond to the models you intend to use. You can find available models and their IDs in the Transkribus WebApp or documentation.

- **Polling Interval:** The polling interval is set to 5 seconds; adjust this as needed based on expected processing times.
- **Error Handling:** The function raises an exception if processing fails. You may want to implement more robust error handling based on your application's requirements.

This implementation consolidates the authentication, image submission, and result retrieval processes into a single function, streamlining the workflow for processing images through the Transkribus Metagrapho API.



```
def __init__(self, username, password):
    self.username = username
    self.password = password
    self.client_id = 'processing-api-client'
    self.auth_url = 'https://account.readcoop.eu/auth/realms/readcoop/protocol/openid-connect'
    self.api_url = 'https://transkribus.eu/processing/v1'
    self.access_token = None
    self.refresh_token = None

def authenticate(self):
    url = f"{self.auth_url}/token"
    payload = {
        'grant_type': 'password',
        'username': self.username,
        'password': self.password,
        'client_id': self.client_id
    }
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}

    response = requests.post(url, data=payload, headers=headers)
    if response.status_code == 200:
        data = response.json()
        self.access_token = data['access_token']
        self.refresh_token = data['refresh_token']
        print("Authentication successful!")
    else:
        print(f"Authentication failed: {response.status_code} - {response.text}")

def refresh_access_token(self):
    if not self.refresh_token:
        raise ValueError("No refresh token available. Authenticate first.")

    url = f"{self.auth_url}/token"
    payload = {
        'grant_type': 'refresh_token',
        'refresh_token': self.refresh_token,
        'client_id': self.client_id
    }
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}
```

```
response = requests.post(url, data=payload, headers=headers)
if response.status_code == 200:
    data = response.json()
    self.access_token = data['access_token']
    self.refresh_token = data['refresh_token']
    print("Token refreshed successfully!")
else:
    print(f"Token refresh failed: {response.status_code} - {response.text}")
```

Thank you for sharing your `__init__`, `authenticate`, and `refresh_access_token` methods. Building upon these, you can integrate the entire process—including authentication, image submission, and result retrieval—into a single method. Here's how you can structure the `call_transkribus_api` function within your class:

`call_transkribus_api(image_path)` Implementation

```
python

import base64
import requests
import time

class TranskribusClient:
    def __init__(self, username, password):
        self.username = username
        self.password = password
        self.client_id = 'processing-api-client'
        self.auth_url = 'https://account.readcoop.eu/auth/realms/readcoop/protocol/openid-connect'

        self.api_url = 'https://transkribus.eu/processing/v1'
        self.access_token = None
        self.refresh_token = None

    def authenticate(self):
        url = f"{self.auth_url}/token"
        payload = {
            'grant_type': 'password',
            'username': self.username,
            'password': self.password,
            'client_id': self.client_id
        }
        headers = {'Content-Type': 'application/x-www-form-urlencoded'}

        response = requests.post(url, data=payload, headers=headers)
        if response.status_code == 200:
            data = response.json()
            self.access_token = data['access_token']
            self.refresh_token = data['refresh_token']
            print("Authentication successful!")
        else:
            print(f"Authentication failed: {response.status_code} - {response.text}")

    def refresh_access_token(self):
        if not self.refresh_token:
            raise ValueError("No refresh token available. Authenticate first.")

        url = f"{self.auth_url}/token"
        payload = {
            'grant_type': 'refresh_token',
            'refresh_token': self.refresh_token,
            'client_id': self.client_id
        }
        headers = {'Content-Type': 'application/x-www-form-urlencoded'}
```

```

response = requests.post(url, data=payload, headers=headers)
if response.status_code == 200:
    data = response.json()
    self.access_token = data['access_token']
    self.refresh_token = data['refresh_token']
    print("Token refreshed successfully!")
else:
    print(f"Token refresh failed: {response.status_code} - {response.text}")

def call_transkribus_api(self, image_path):
    # Ensure authentication
    if not self.access_token:
        self.authenticate()

    # Model IDs (replace with appropriate IDs for your use case)
    LINE_MODEL_ID = 51962 # Default line detection model ID
    HTR_MODEL_ID = 51170 # Example: Transkribus Print M1

    # Encode Image to Base64
    with open(image_path, 'rb') as image_file:
        image_bytes = image_file.read()
    image_base64 = base64.b64encode(image_bytes).decode('utf-8')

    # Submit Image for Processing
    process_url = f"{self.api_url}/processes"
    process_headers = {
        'Authorization': f'Bearer {self.access_token}',
        'Content-Type': 'application/json',
        'Accept': 'application/json'
    }
    process_payload = {
        "config": {
            "textRecognition": {
                "htrId": HTR_MODEL_ID
            },
            "lineDetection": {
                "modelId": LINE_MODEL_ID
            }
        },
        "image": {
            "base64": image_base64
        }
    }
    process_response = requests.post(process_url, json=process_payload,
headers=process_headers)
    if process_response.status_code != 200:
        print(f"Image submission failed: {process_response.status_code} - {process_response.text}")
        return None
    process_id = process_response.json()['processId']
    print(f"Image submitted successfully! Process ID: {process_id}")

    # Poll for Processing Status and Retrieve Result
    status_url = f"{self.api_url}/processes/{process_id}"
    while True:
        status_response = requests.get(status_url, headers={'Authorization': f'Bearer {self.access_token}'})
        if status_response.status_code != 200:
            print(f"Status check failed: {status_response.status_code} - {status_response.text}")
            return None
        status_data = status_response.json()
        status = status_data.get('status')
        if status == 'FINISHED':
            recognized_text = status_data.get('result', {}).get('text', '')
            return recognized_text
        elif status == 'FAILED':
            print("Processing failed.")
            return None
        time.sleep(5) # Wait before polling again

```

Notes:

- **Model IDs:** Ensure that `LINE_MODEL_ID` and `HTR_MODEL_ID` correspond to the models you intend to use. You can find available models and their IDs in the Transkribus WebApp or documentation.
- **Error Handling:** The function includes basic error handling for authentication, image submission, and status checking. You may want to implement more robust error handling based on your application's requirements.
- **Polling Interval:** The polling interval is set to 5 seconds; adjust this as needed based on expected processing times.

This implementation consolidates the authentication, image submission, and result retrieval processes into a single method within your class, streamlining the workflow for processing images through the Transkribus Metagrapho API.